Deep Learning for Lung Cancer Detection: Using Pretrained and Custom Models on Low-Volume Medical Imaging Data: CS 7643

Ajay Chopra, Jeremy Devin, Fernando Mendez Campos
Georgia Institute of Technology
achopra74@gatech.edu, jdevin3@gatech.edu, fmc6@gatech.edu

Abstract

A major difficulty in applying deep learning to medical imaging is lack of access to large amounts of data. In this paper, we utilize both pretrained and custom models to identify lung cancer in patients using the IQ-OTH/NCCD dataset, which contains approximately 3600 total images. We compare the results of 5 models pretrained on ImageNet (3 convolutional architectures and 2 vision transformers), and seven custom model architectures (four convolutional networks and three vision transformers). With both custom and pretrained models, we were able to surpass a recent benchmark set by researchers on this dataset of 94.38%. Moreover, in certain experiments we were able to surpass this benchmark using only 50% of the training data used in the original benchmark. Therefore, we introduce several promising methods that can be used to classify medical images from extremely low-volume datasets, which could be extremely beneficial in furthering the application of deep learning to the domain of medical imaging.

1. Introduction

An issue that has consistently hampered researchers trying to apply deep learning to medical images is limited access to data [3]. Unlike data for other image classification tasks that may be freely available on the Internet, medical imaging data is usually distributed throughout numerous institutions and is not publicly available [11]. Moreover, it is extremely time-consuming and expensive to collect [11].

Therefore, classification techniques which deliver high performance in a low-volume data setting are needed. Transfer learning - a process by which models are trained on large datasets and then fine-tuned on the target task - has proven successful for certain medical imaging applications. In 2021, Agarwal et al. used transfer learning to identify signs of Alzheimer's from brain MRIs [1]. During the COVID-19 pandemic, several researchers utilized transfer learning to detect COVID-19 via chest X-ray images [13]. Additionally, there has been much research on developing custom deep networks which can perform well in a low-volume data setting. Mavaie et al. showed success in using such an approach to classify low-volume, high-dimensional biological data [12].

In this paper, we further this research by comparing several pretrained and custom models on the task of lung cancer detection. We used the IQ-OTH / NCCD lung cancer dataset, which is a collection of slices of CT scans of patients - some of whom had been diagnosed with lung cancer, some of whom had benign lung tumors, and some who were healthy patients [8]. It was collected by the Iraq-Oncology Teaching Hospital and the National Center for Cancer Diseases over the course of three months. Each slice is marked as benign, malignant, or normal. The original data set contained 120 benign, 561 malignant and 416 normal slices. To increase the amount of data and evenly distribute samples across all three classes, the dataset publishers performed several augmentations (flips, rotations, color jitters, Gaussian blur) so that the published dataset contains 1200 examples of each class [8]. Example images from the dataset (before preprocessing) can be seen in Figure 1.

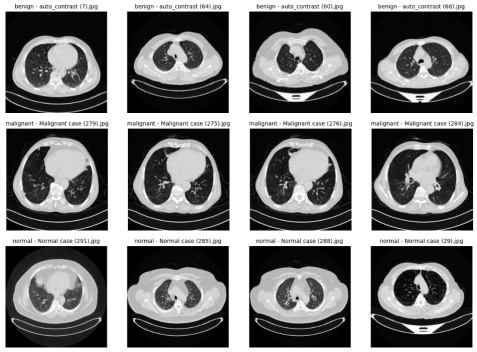

Figure 1. Four examples from each class of our dataset.

A benchmark for this dataset was published by Al-Huseiny et al. in 2021. Using a pretrained GoogLeNet, they achieved 94.38% classification accuracy on the test

data after fine-tuning [2]. Other researchers have performed similar methods using AlexNet and some have utilized non DL methods such as SVM [10, 9], achieving similar performance.

Our goal with this paper was to achieve higher accuracy than the benchmark, both with pretrained and custom models. As is discussed in our results, we achieved this goal. Moreover, in certain cases, we were able to beat the benchmark using only 50% of the training data used in the benchmark paper. Therefore, we introduce multiple promising methods which may be used to perform classification on extremely small datasets. We believe this paper can advance the application of deep networks in the medical image classification domain.

2. Approach

Unless otherwise noted, we used PyTorch for all data preprocessing, model creation, training and analysis. For data preprocessing, we took inspiration from this notebook. More specific attributions can be found in our code comments. Our code is attached as a supplemental in our submission.

2.1. Data Preprocessing

The first preprocessing step we performed was converting the color space of the images from BGR (the default color space used by OpenCV), to RGB, which is the color space of ImageNet images. One problem we anticipated was the lack of uniform size among the provided images. For instance, some images were of size 224×224, while others were 530×530. To prevent training issues due to dimension mismatch, we decided to resize all images to be 224×224. We did this using pixel area relation, since it is best for shrinking images. This is one key area in which we differed from the starter code. Lastly, we did not want large pixel values leading to instability during training, so we normalized all pixel values to be within a range of 0-1. The starter code performed additional normalization to place pixels in a range between -1 and 1, but we did not perform this additional step, since we did not believe negative pixel values carried any additional significance and perhaps could lead to training instability.

In terms of splitting the data into training, validation and test data, we used a 70%, 15%, 15% split, mainly because this is what was used in the benchmark paper [2].

2.2. Transfer Learning on Pre-trained Models

We compared the performance of three pretrained models: ResNet 50, VGG 13, and SqueezeNet [6, 7, 4]. ResNet and VGG were chosen because they are both deep convolutional neural networks, but differ in that ResNet employs skip connections, whereas VGG does not [6, 7]. It has been shown that when training these networks from scratch on a large dataset such as ImageNet, these skip connections help to mitigate the vanishing/exploding gradients problem, which occurs when gradients become exponentially smaller/larger as they are backpropogated from the output layer to the input layer of the network [6]. We wanted to test if this also held true in a fine-tuning context and led to better performance from ResNet than from VGG.

SqueezeNet was chosen because it is one of the smallest widely used architectures trained on ImageNet (in terms of trainable parameters) and would serve as an interesting point of comparison to the larger models.

The process of pretraining was roughly similar for each model. All models were pretrained on the ImageNet dataset [5]. For the first round of fine-tuning, we froze every layer besides the final classification layer. For ResNet and VGG, this was a fully connected linear layer whose output dimension was equal to the number of classes. For SqueezeNet, this was a 2D convolutional layer with output dimension equal to the number of classes. Then for the second round, we unfroze the last few layers of each network. For VGG, this was all linear layers after feature extraction. For ResNet, this was the entire final block of convolutional layers (referred to as "layer4" in the PyTorch model). For SqueezeNet, it was the last "fire module" i.e. the last squeeze and expand layer.

For each model, cross entropy loss was used as the loss function and Adam was used as the optimizer. In terms of hyperparameters, in order to keep the procedure consistent among the models, we only tuned the learning rate in our optimizer. The learning rate was tuned via grid search for each model. For all models, we found that smaller learning rates (0.0001 - 0.0005) performed best. In general we found that whichever learning rate we used for the first round of fine-tuning (e.g. 0.0001), multiplying that learning rate by 0.1 (0.00001) gave the best results for round two. The fact that smaller learning rates gave the best performance makes sense, given that during fine-tuning we only want to make small changes to the model weights, since they are already mostly optimal.

2.3. Training Custom Convolutional Networks

The second phase of exploration was to train custom CNNs for comparison to existing benchmarks. Our goal was to design models that match or outperform the benchmark of 94.38% classification accuracy while minimizing the number of trainable parameters in each model. To do this, we started with a model comprising of three convolutional blocks in the feature extractor, inspired by the Enhanced CNN model architecture designed by Shatnawi et al. [14]. The model has a classifier with two fully-connected layers to process flattened features. From this model, three similar models were developed to preserve accuracy while reducing the number of trainable parameters as much as

possible. The models are explained in detail below.

CNN1 serves as the starting architecture, comprising of three convolutional blocks with channel dimensions (64→32→32) and two max-pooling layers (reducing spatial resolution from 224 × 224 to 56 × 56). Each block integrates batch normalization (BatchNorm), LeakyReLU activation (α=0.1), and incremental dropout (0.2, 0.3, and 0.4) to regularize intermediate features. The classifier consists of two fully connected layers (256→128 units) with BatchNorm and dropout, processing flattened 32 × 56 × 56 features (100,352 dimensions).

CNN2 introduces spatial and channel efficiency by reducing the initial block's width (64→32 channels) and adding a third max-pooling layer (56 × 56 → 28 × 28) in the last convolutional block. This compresses the classifier input to 25,088 dimensions. A bottleneck layer is added to distill features in the classifier. These modifications require about 1/8 of the parameters to be trained as CNN1.

CNN3 further optimizes the classifier by replacing the multi-layer design with a single 64-unit hidden layer. The removal of one BatchNorm layer and reduction in linear units (25,088→64) cuts the number of parameters in half from CNN2.

CNN4 represents an extreme bottleneck variant, collapsing the classifier to a 32-unit linear layer followed by ReLU. This architecture emphasizes minimalism and inference speed, suitable for resource-constrained environments. We noted that its reduced capacity may benefit from enhanced data augmentation or regularization to maintain robustness. With half the parameters as CNN3, it represents a significant reduction from CNN1 and serves as a suitable minimal network for the purpose of our investigation. A visual representation of CNN4 is shown in Figure 2.

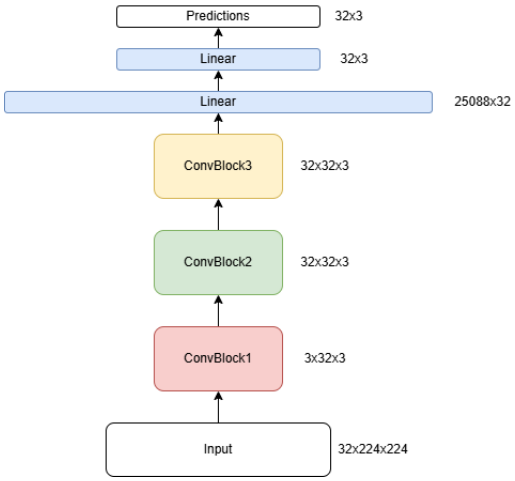

Figure 2. Diagram of CNN4.

After developing these network architectures, it was a critical next step to find the best optimizer, learning rate, weight decay, and number of epochs to best compare each CNN. Optimizers were chosen between Adam and AdamW, learning rates were tested within a set of standard rates (0.0001, 0.001, 0.01), weight decays were tested at 0, 0.001, and 0.01, and number of epochs were chosen 10-30. As tuning these hyperparameters is not the focus of this section, a suitable choice for each was made and applied to all custom CNN models to achieve good accuracy while enabling comparison between number of parameters versus validation accuracy.

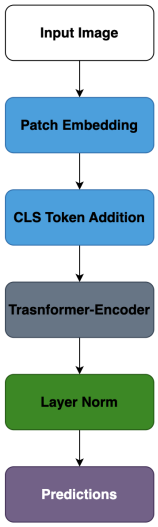

Figure 3. Diagram of custom ViT architecture

2.4. Training Custom Vision Transformer Models (ViTs)

In addition to CNN architectures, we wanted to compare the performance of a transformer architecture on this dataset. We trained and compared five different models- three custom transformers (TinyViT, MediumViT, LargeViT) and two pre-trained transformers (ViT-Small, ViT-Base) to determine the tradeoffs between different design decisions. As with the CNN, our goal was to beat the established benchmark of 94.38 percent accuracy. Below is an illustration of the architecture for the custom ViTs:

The rationale behind the three custom models- TinyViT, MediumViT, and LargeViT was to determine whether increasing model capacity improved performance or not. The pre-trained models were then compared against the custom models to determine whether transfer learning would be effective for this task. The results of all five models are explained in the next section.

3. Experiments and Results

For our experiments, we considered a successful model to be any model (either pretrained or custom) that achieved

higher accuracy on the validation data than the benchmark set by Al-Huseiny et al.

3.1. Results of Pretrained Models

From Table 1, we can see that of our pretrained models, ResNet performed the best in terms of accuracy on the test data, beating the benchmark set by the pretrained GoogLeNet. It narrowly outperformed VGG and greatly outperformed the SqueezeNet. Additionally, this held even as the amount of data used for training decreased. We can see that even when using only 50% of the provided training data, pretrained ResNet was still able to outperform the benchmark, as well as the other pretrained networks.

We claim that the ResNet outperformed VGG because the skip connections in ResNet successfully mitigated the vanishing gradients problem. Evidence for this can be shown in Figure 4, in which we display the gradient activations for the first and last fine-tuned layers of both ResNet and VGG half way through (i.e. at epoch 5) the fine tuning process. While the gradients at the final layer both show a wide distribution around 0, we can see that for the first layers, the VGG gradients all collapse to near 0. We do see some collapse in ResNet as well, but not nearly to the degree of VGG. This is a clear indicator of vanishing gradients even within our pretrained VGG network.

Examining the loss curves in Figure 5, we can see that although ResNet outperformed VGG in terms of validation accuracy, it also exhibited more signs of overfitting. This was somewhat surprising, given that VGG is a deeper and more complex network than ResNet (at least in terms of the number of parameters trained), and so one would expect it to overfit on a small dataset. One possible explanation might be that the relatively higher learning rate chosen for ResNet (0.00005 vs. 0.00001) exacerbated the overfitting, despite still giving good performance on test data.

As for why both VGG and ResNet outperformed SqueezeNet, we claim that SqueezeNet's smaller model size prevents it from being as accurate as larger models on image classification tasks. In other words, SqueezeNet exhibits modeling error, which occurs when a model cannot estimate a function within a given accuracy no matter how much it is optimized. Though it is difficult to show this via visualizations, we cite SqueezeNet's lower performance on ImageNet (80.42% accuracy at 5 epochs comapred to 95.434% for ResNet) as evidence of its lower performance as compared to larger models [4, 6].

3.2. Results of Custom Convolutional Networks

For each CNN, hyperparameters are set as follows: optimizer was chosen as AdamW with learning rate of 0.0001 and weight decay of 0.01. Each CNN was trained for 20 epochs. The number of parameters and resulting validation accuracy of each CNN is shown in Table 2, and loss curves

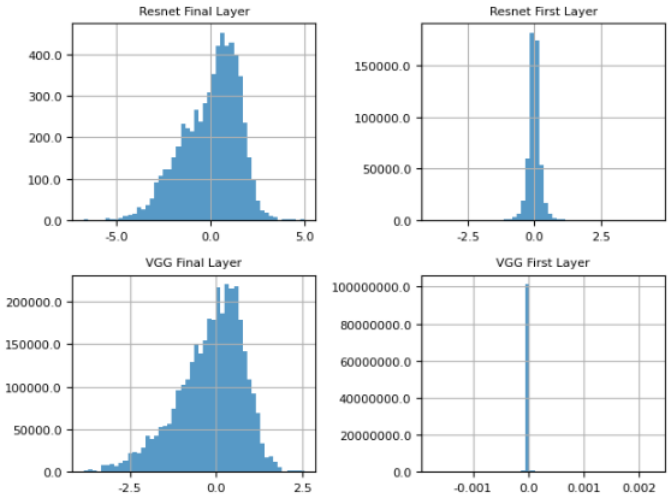

Figure 4. Gradient distributions for the first and last fine-tuned layers for both ResNet and VGG. The x-axis represents the values of the gradients for the layer at the final epoch and the y-axis is the frequency.

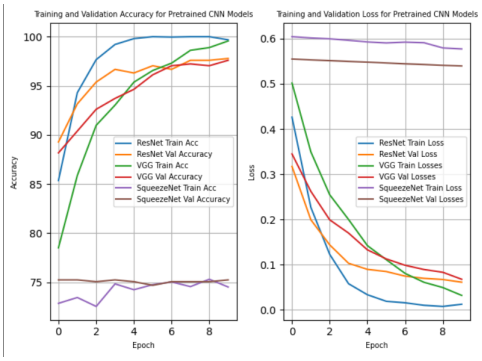

Figure 5. Training curves for the pretrained CNNs.

calculated with cross-entropy are shown in Figure 5.

To start, the validation accuracy of CNN1 is higher than the benchmark of 94.38% from Al-Huseiny et al. [2]. This result is already noteworthy, as the pretrained GoogLeNet produced state-of-the-art results in the 2021 paper. It's worth noting that GoogLeNet is a convolutional neural network designed to classify objects among 1000 categories, so it is possible that our custom CNN trained from scratch could better optimize to the 3 categories (benign, malignant, and normal). Regardless, 25.8 million trainable parameters is much fewer than VGG networks and around the same number as ResNet 50.

After CNN1 was trained with success, CNN2 was trained and was not able to beat the benchmark. Initially, this result made sense, as CNN2 has only roughly 1/8 of the parameters of CNN1. Regardless, the experiment was continued with the other custom CNNs. After training the

Model Name	Parameters Trained	Test Accuracy (50% train data)	Test Accuracy (full train data)
ResNet 50	14.9M	97.23%	97.41%
VGG 13	119.5M	94.77%	95.29%
SqueezeNet	1539	74.46%	80.04%

Table 1. Results of pretrained CNN models on lung cancer detection.

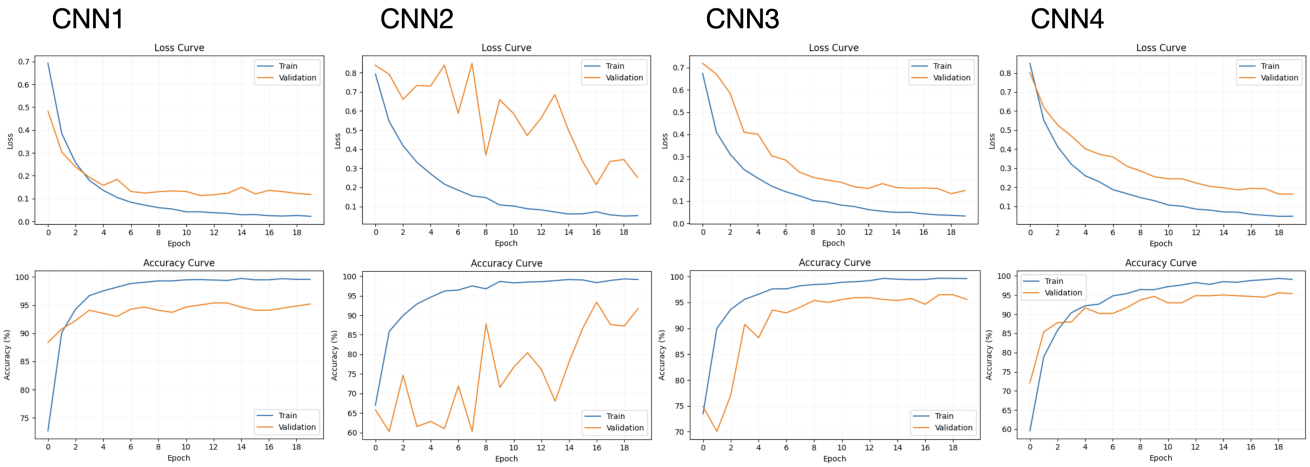

Figure 6. Loss and accuracy curves for each custom CNN.

Model Name	Parameters Trained	Validation Accuracy
CNN1	25.75 M	95.19%
CNN2	3.24 M	92.05%
CNN3	1.63 M	95.01%
CNN4	0.82 M	94.82%

Table 2. Results of training custom CNNs.

other CNNs, it became apparent that CNN2's architecture led to a strangely high variance in validation loss and accuracy across epochs. This is visually apparent in the loss curves shown in Figure 5. The change in architectures between CNN2 and CNN3 could explain why CNN2 shows the least consistent validation results despite having more trainable parameters than CNN3 and CNN4.

Surprisingly, CNN3 resulted in a validation accuracy near that of CNN1, with only 1/16 of the trainable parameters. This could be explained by the change in network architecture, particularly the change in the classifier from a multi-layer design to a single 64-unit layer. Further tests would be required to pinpoint the explanation for this rebound in accuracy, but the immediate goal was to create a CNN with minimal parameters that still performs well.

CNN4, despite having around 1/32 of the trainable parameters as CNN1, performed almost as well, with validation accuracies within half a percentage point from each other. CNN4 was therefore able to beat the benchmark of 94.38% with under a million parameters. This result speaks to the impact of CNN architecture on the performance of specific classification tasks.

3.3. Results of Transformer Architecture

The results of our transformer ViT models can be found on the next page in Table 3. As a reminder, we trained five different Vision Transformer (ViT) architectures on the lung cancer classification dataset: three custom models (TinyViT, MediumViT, LargeViT) and two pre-trained models (ViT-Small and ViT-Base from the TIMM library). The highest accuracy came from the pre-trained Small ViT with an accuracy of 94.48%. It was the only model which beat the benchmark established with this data set (94.38%). The next best performer was the custom TinyViT at 92.40% accuracy. It is interesting to note that the smaller models performed significantly better than the larger models. The smaller ViTs generalized better on this dataset compared to larger ViTs, perhaps due to the relatively small dataset size (3600 images).

Looking at the loss and accuracy curves, we also notice something interesting. The larger models (and particularly the pre-trained ViT Base timm model) experienced low accuracy and high loss in earlier epochs yet improved significantly as training went on. Perhaps this is an indication that given more training epochs, their performance might have improved even further. The best performer in terms of loss and accuracy from earlier epochs all the way to the final

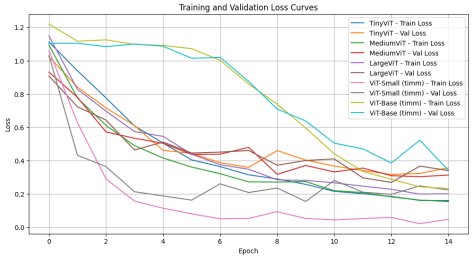

Figure 7. Loss curves for ViT models

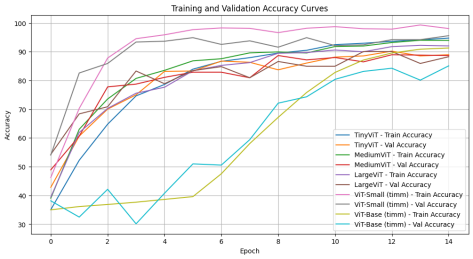

Figure 8. Accuracy curves for ViT models

Model Name	Validation Accuracy
Custom TinyViT	92.40%
Custom MediumViT	89.78%
Custom LargeViT	83.56%
Pre-trained ViT-Small (timm)	94.48%
Pre-trained ViT-Base (timm)	86.60%

Table 3. Results of training Vision Transformer variants.

epochs was the pre-trained small ViT- again showing that for this relatively small dataset, using a pre-trained model can be highly effective. Overall, however, the ViT architectures did not perform as well as the CNNs.

4. Discussion

4.1. Challenges and Adjustments

Our first challenge was handling limited memory in our local development environments. Even though the lung cancer dataset is originally 178 MB on disk, the preprocessing step requires more than 16 GB RAM to complete. Thus, we opted to develop and test using Google Colab, where we could use High-RAM instances (over 50 GB) on the Google Compute Engine backend.

Another challenge that arose was creating a general fine-tuning procedure for all of our pretrained models. On one hand, we wanted the procedure to be standardized across all model architectures in order to provide an accurate comparison between them. However, the model architectures that we tested varied greatly, and so we had to think carefully about which parameters of each model to unfreeze during fine tuning. One idea we had was to unfreeze the same number of parameters in each model, but this proved to be incompatible with the differing architectures. Therefore, we settled on unfreezing the last 3-4 layers of each network.

One challenge that came up during the design and testing of custom CNNs was the speed at which we could adjust the models and hyperparameters. Initially, we chose a common learning rate (0.0001) for training, used the Adam optimizer, and trained models for 20 epochs. This proved to be slow, as we were testing many iterations of modified CNN architectures before narrowing them down to the selected four. As such, we adjusted the training time down to 5 epochs for the exploratory part of the process. This sped up training time, but it increased the risk of overfitting, as some models demonstrated a decrease in validation accuracy after 5 to 7 epochs. After finding suitable CNNs for further adjustments, we tested a standard set of hyperparameters across the networks, as noted in the Approach section. We increased the number of epochs back to 20 to get more fine-grained results while tuning. This proved to be time-intensive, but we determined the extra time was worth the extra confidence in the results.

4.2. Successes and Opportunities

Across our experiments, we found several models that performed better than the benchmark from Al-Huseiny et al. Moreover, several of these models were able to outperform the pretrained GoogLeNet in terms of validation accuracy while only using half the data used to fine-tune the GoogLeNet. In addition, we showed that the problem of classifying images in a low-volume data setting can be accomplished through both pretrained and custom convolutional networks and vision transformers. This versatility makes these findings even more applicable in a real world scenario.

One area for future exploration would be testing the implementation of the minimal custom network (CNN4) on a resource-constrained device, such as a mobile phone. Such a minimal network would benefit greatly from highly regularized data, so it would be worth investigating how our trained models perform on new data outside the dataset. For example, could be a system developed to take new CT scan images of lungs and crop or reshape them to the same specifications of the the IQ-OTH / NCCD lung cancer dataset.

Another area of exploration outside this project is the ethical implementation of AI in medical imaging decisions. Even with a validation accuracy of over 95% on our best models, any real-world deployment should require rigorous clinical validation, transparent error analysis, and clear accountability frameworks. AI should augment, not replace, physician judgment, particularly in high-stakes scenarios like cancer diagnosis, where false positives and negatives carry profound consequences.

Student Name	Contributed Aspects	Details
Ajay Chopra	Data Preprocessing and Pre-Trained CNN Fine-Tuning	Preprocessed the images for the project and performed fine-tuning on pre-trained CNN models. Analyzed differences in gradient distributions between model architectures during pre training. Developed initial design for custom CNN (that was improved upon by Jeremy)
Fernando Mendez Campos	Custom and Pre-Trained Vision Transformer Comparison	Coded and trained 5 different transformer models (3 custom ones and 2 pre-trained ones) to compare their performance on the lung cancer dataset. Analyzed results and loss/accuracy curves for all 5 models.
Jeremy Devin	Custom CNNs	Designed the architecture of custom CNNs, implemented and trained them, and analyzed the results. Analyzed effect of trainable parameters on validation accuracy.

Table 4. Contributions of team members.

References

[1] Deevyankar Agarwal. Transfer learning for alzheimer's disease through neuroimaging biomarkers: A systematic review. *Sensors*, 2021. 1

[2] Muayed Al-Huseiny. Transfer learning with googlenet for detection of lung cancer. *Indonesian Journal of Electrical Engineering and Computer Science*, 2021. 2, 4

[3] T. Dhar. Challenges of deep learning in medical image analysis - improving explainability and trust. *IEEE Transactions on Technology and Society*, 2023. 1

[4] Forrest Iandola et al. Squeezenet: Alexnet-level accuracy with 50x fewer prameters and ¡0.5mb model size. *ICLR 2017*, 2017. 2, 4

[5] Jia Dent et al. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 2

[6] Kaiming He et al. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2, 4

[7] Shuying Liu et al. Very deep convolutional neural network based image classification using small training sample size. *Asian Conference on Pattern Recognition*, 2015. 2

[8] Alyasriy Hamdalla. The iq-othnccd lung cancer dataset. *Mendeley Data*, 2021. 1

[9] E. A. Khalil. Diagnosis of lung cancer based on ct scans using cnn. *IOP Conference Series: Materials Science and Engineering*, 2020. 2

[10] E. A. Khalil. Evaluation of svm performance in the detection of lung cancer in marked ct scan dataset. *Indonesian Journal of Electrical Engineering and Computer Science*, 2021. 2

[11] Juan Li. Medical image identification methods: A review. *Computers in Biology and Medicine*, 2024. 1

[12] Pegah Mavaie. Hybrid deep learning approach to improve classification of low-volume high-dimensional data. *BMC Bioinformatics*, 2023. 1

[13] Mamunur Rahaman. Identification of covid-19 samples from chest x-ray images using deep learning: A comparison of transfer learning approaches. *Journal of X-Ray Science and Technology*, 2020. 1

[14] Mohammad Q. Shatnawi, Qusai Abuein, and Romesaa Al-Quraan. Deep learning-based approach to diagnose lung cancer using ct-scan images. *Intelligence-Based Medicine*, 11(100188), 2025. 2